VERİ TABANI SORU ÖRNEKLERİ -4,5 (RA ve SQL)

Examples on COMPANY DB :

Figure 5.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

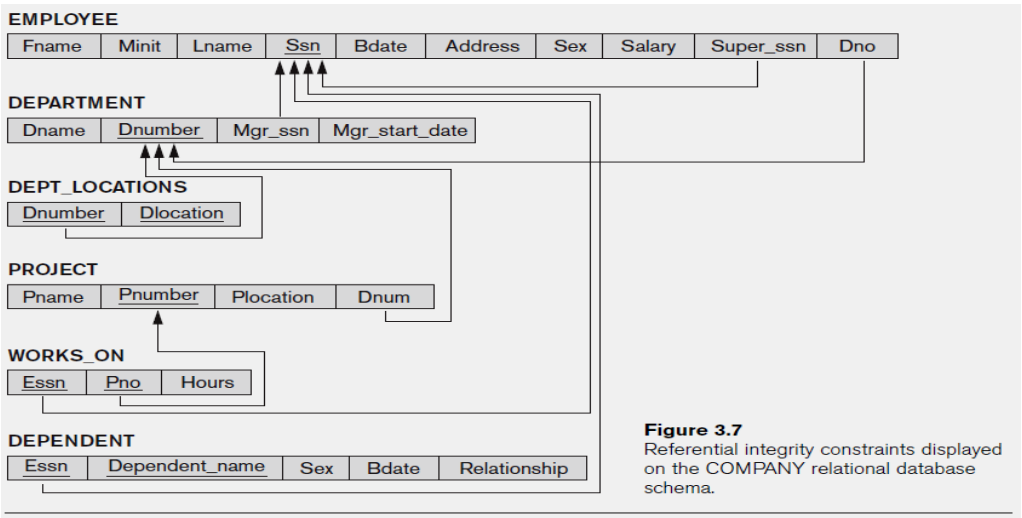


Figure 3.7
Referential integrity constraints displayed on the COMPANY relational database schema.

- List of employee names and also the name of departments they manage if exist.

TEMP ← (EMPLOYEE $\bowtie_{SSN=MGRSSN}$ DEPARTMENT)

RESULT ← $\pi_{FNAME, MINIT, LNAME, DNAME}$(TEMP)

RESULT

Fname	Minit	Lname	Dname
John	B	Smith	NULL
Franklin	T	Wong	Research
Alicia	J	Zelaya	NULL
Jennifer	S	Wallace	Administration
Ramesh	K	Narayan	NULL
Joyce	A	English	NULL
Ahmad	V	Jabbar	NULL
James	E	Borg	Headquarters

Figure 6.12
The result of a LEFT OUTER JOIN operation.

Q1: Retrieve the name and address of all employees who work for the 'Research' department.

$RESEARCH\_DEPT = \sigma_{DNAME='Research'}(DEPARTMENT)$

$RESEARCH\_EMPS = (RESEARCH\_DEPT \bowtie_{DNUMBER=DNO} EMPLOYEE)$

$RESULT = \pi_{FNAME, LNAME, ADDRESS}(RESEARCH\_EMPS)$

Q2:For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

$STAFFORD\_PROJS \leftarrow \sigma_{PLOCATION='STAFFORD'}(PROJECT)$

$CONTR\_DEPT \leftarrow (STAFFORD\_PROJS \bowtie_{DNUM=DNUMBER} DEPARTMENT)$

$PROJ\_DEPT\_MGR \leftarrow (CONTR\_DEPT \bowtie_{MGRSSN=SSN} EMPLOYEE)$

$RESULT \leftarrow \pi_{PNUMBER, DNUM, LNAME, ADDRESS, BDATE}(PROJ\_DEPT\_MGR)$

Q3.) Find the names of employees who work on all the projects controlled by department number 5.

$DEPT5\_PROJS \leftarrow \rho_{PNO}(\pi_{PNUMBER}(\sigma_{DNUM=5}(PROJECT)))$

$EMP\_PROJ \leftarrow \rho_{SSN, PNO}(\pi_{ESSN, PNO}(WORKS\_ON))$

$RESULT\_EMP\_SSNS \leftarrow EMP\_PROJ \div DEPT5\_PROJS$

$RESULT \leftarrow \pi_{LNAME, FNAME}(RESULT\_EMP\_SSNS * EMPLOYEE)$

Q4.) Make a list of project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

$SMITHS \leftarrow \rho_{ESSN}(\pi_{SSN}(\sigma_{NAME='SMITH'}(EMPLOYEE))$

$SMITH\_WORKER\_PROJ \leftarrow \pi_{PNO}(WORKS\_ON * SMITHS)$

$MGRS \leftarrow \pi_{LNAME, DNUMBER}(EMPLOYEE \bowtie_{SSN=MGRSSN} DEPARTMENT)$

$SMITH\_MANAGED\_DEPTS \leftarrow \rho_{DNUM}(\pi_{DNUMBER}(\sigma_{NAME='SMITH'}(MGRS)))$

$SMITH\_MGR\_PROJS \leftarrow \rho_{PNO}(\pi_{PNUMBER}(SMITH\_MANAGED\_DEPTS * PROJECT))$

$RESULT \leftarrow (SMITH\_WORKER\_PROJS \cup SMITH\_MGR\_PROJS)$

Q5.) List the names of all employees with two or more dependents. (We assume that dependents of the same employee have distinct DEPENDENT_NAME values.)

$T1 \leftarrow \rho_{SSN,\ NO\_OF\_DEPTS}\ (ESSN\ \mathcal{F}_{COUNT(DEPENDENT\_NAME)}\ DEPENDENT)$

$T2 \leftarrow \sigma_{NO\_OF\_DEPS>2}\ (T1)$

$RESULT \leftarrow \pi_{LNAME,\ FNAME}(T2\ *EMPLOYEE)$

Q6: Retrieve the names of employees who have no dependents.

$ALL\_EMPS = \pi_{SSN}(EMPLOYEE)$

$EMPS\_WITH\_DEPS = \rho_{SSN}\ (\ \pi_{ESSN}(DEPENDENT)\ )$

$EMPS\_WITHOUT\_DEPS\ =\ (ALL\_EMPS\ -\ EMPS\_WITH\_DEPS)$

$RESULT\ =\ \pi_{LNAME,\ FNAME}\ (EMPS\_WITHOUT\_DEPS\ *\ EMPLOYEE)$

Q7)List the names of managers who have at least one dependent.

$MGRS(SSN) \leftarrow \pi_{MGRSSN}(DEPARTMENT)$

$EMPS\_WITH\_DEPS\ (_{SSN}) \leftarrow \pi_{ESSN}\ (DEPENDENT)$

$MGRS\_WITH\_DEPS \leftarrow (MGRS\ \cap\ EMPS\_WITH\_DEPS)$

$RESULT \leftarrow \pi_{LNAME,\ FNAME}\ (MGRS\_WITH\_DEPS\ *\ EMPLOYEE)$

Q8) Aile efradından (dependent), kendisiyle aynı ilk isimde olan işçilerin isimlerinin bulunması…

$EMPDEP \leftarrow EMPLOYEE\ \bowtie_{SSN=ESSN}\ DEPENDENT$

$R \leftarrow \pi_{FNAME}\ \sigma_{FNAME=DEPENDENT\_NAME}\ (EMPDEP)$

Q9) 'Franklin Wong' isimli amirin idaresinde çalışan bütün işçilerin isimlerinin bulunması…

$R1 \leftarrow \pi_{SSN}\ \sigma_{FNAME='FRANKLIN\ WANG'}(EMPLOYEE)$

$R \leftarrow \pi_{FNAME,LNAME}\ (EMPLOYEE\ \bowtie_{SUPERSSN=SSN}\ R1)$

Q10) Her proje için, projenin ismi ve bütün personelin bu proje için haftada harcadığı toplam saatin bulunması…

$R1 \leftarrow \pi_{PNUMBER,PNAME,HOURS}\ (PROJECT\ \bowtie_{PNUMBER=PNO}\ WORKS\_ON)$

$R \leftarrow {}_{PNUMBER}\mathcal{F}_{SUM(HOURS)}(R1)$

Q11) Projelerin hepsinde yer alan işçilerin isimlerinin bulunması…

$$ALL\_PROJS \leftarrow \pi_{PNUMBER}(PROJECT))$$

$$EMP\_PROJ \leftarrow \pi_{ESSN,\ PNO}\ (WORKS\_ON)$$

$$R1 \leftarrow EMP\_PROJ \div ALL\_PROJS$$

$$R \leftarrow \sigma_{FNAME}\ (R1 \bowtie_{ESSN=SSN}\ EMPLOYEE)$$

Q12) Hiç bir projede çalışmayan işçilerin isimlerinin bulunması…

$$R1 \leftarrow \pi_{SSN}(EMPLOYEE) - \pi_{ESSN}(WORKS\_ON)$$

$$R \leftarrow R1 * EMPLOYEE$$

Q13) Her departman için departmanın ismi ve bu departmanda çalışan işçilerin maaşlarının ortalamasının bulunması…

$$R1 \leftarrow DEPARTMENT \bowtie_{DNUMBER=DNO}\ EMPLOYEE$$

$$R \leftarrow {}_{DNO}\mathcal{F}_{AVERAGE(SALARY)}(R1)$$

Q14) "Houston"daki bir projede çalışan fakat bağlı olduğu departmanın "Houston"da bir şubesi olmayan işçilerin isimlerinin ve adreslerinin bulunması…

$$H \leftarrow \rho_{PNO}\ \pi_{PNUMBER}\ \sigma_{PLOCATION='HOUSTON'}\ (PROJECT)$$

$$HSSN \leftarrow \rho_{SSN}\ \pi_{ESSN}\ (H*WORKS\_ON)$$

$$R \leftarrow \pi_{PFNAME,ADRES}\ \sigma_{DLOCATION\ !='Houston'}\ ((HSSN*EMPLOYEE) \bowtie_{DNUMBER=DNO}\ DEPT\_LOCATIONS)$$

Q15) Hiç bir aile efradı olmayan departman yöneticilerinin (managers) soyadlarının bulunması…

$$MGRS \leftarrow \pi_{MGRSSN} DEPARTMENT$$

$$NODEPTMAN \leftarrow \rho_{SSN} MGRS - \pi_{ESSN}\ (DEPENDENT)$$

$$R \leftarrow \pi_{LNAME}\ (EMPLOYEE*NODEPTMAN)$$

Query 0. Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.

```
Q0:   SELECT   Bdate, Address
      FROM     EMPLOYEE
      WHERE    Fname='John' AND Minit='B' AND Lname='Smith';
```

Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

```
Q1:   SELECT   Fname, Lname, Address
      FROM     EMPLOYEE, DEPARTMENT
      WHERE    Dname='Research' AND Dnumber=Dno;
```

Retrieve all the attribute values of the selected tuples

```
Q1C:  SELECT  *
      FROM    EMPLOYEE
      WHERE   Dno=5;

Q1D:  SELECT  *
      FROM    EMPLOYEE, DEPARTMENT
      WHERE   Dname='Research' AND Dno=Dnumber;
```

Figure 4.3
Results of SQL queries when applied to the COMPANY database state shown in Figure 3.6. (a) Q0. (b) Q1. (c) Q2. (d) Q8. (e) Q9. (f) Q10. (g) Q1C.

(a)

Bdate	Address
1965-01-09	731Fondren, Houston, TX

(b)

Fname	Lname	Address
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX

(c)

Pnumber	Dnum	Lname	Address	Bdate
10	4	Wallace	291Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291Berry, Bellaire, TX	1941-06-20

Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

```
Q2:   SELECT   Pnumber, Dnum, Lname, Address, Bdate
      FROM     PROJECT, DEPARTMENT, EMPLOYEE
      WHERE    Dnum=Dnumber AND Mgr_ssn=Ssn AND
               Plocation='Stafford';
```

Queries 9 and 10. Select all EMPLOYEE Ssns (Q9) and all combinations of EMPLOYEE Ssn and DEPARTMENT Dname (Q10) in the database.

```
Q9:   SELECT   Ssn
      FROM     EMPLOYEE;

Q10:  SELECT   Ssn, Dname
      FROM     EMPLOYEE, DEPARTMENT;
```

Query 4. Make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

```
Q4A:  ( SELECT   DISTINCT Pnumber
        FROM     PROJECT, DEPARTMENT, EMPLOYEE
        WHERE    Dnum=Dnumber AND Mgr_ssn=Ssn
                 AND Lname='Smith' )
        UNION
      ( SELECT   DISTINCT Pnumber
        FROM     PROJECT, WORKS_ON, EMPLOYEE
        WHERE    Pnumber=Pno AND Essn=Ssn
                 AND Lname='Smith' );
```

```
Q4A:  SELECT   DISTINCT Pnumber
      FROM     PROJECT
      WHERE    Pnumber IN
               ( SELECT   Pnumber
                 FROM     PROJECT, DEPARTMENT, EMPLOYEE
                 WHERE    Dnum=Dnumber AND
                          Mgr_ssn=Ssn AND Lname='Smith' )
               OR
               Pnumber IN
               ( SELECT   Pno
                 FROM     WORKS_ON, EMPLOYEE
                 WHERE    Essn=Ssn AND Lname='Smith' );
```

Query 11. Retrieve the salary of every employee (Q11) and all distinct salary values (Q11A).

```
Q11:  SELECT   ALL Salary
      FROM     EMPLOYEE;

Q11A: SELECT   DISTINCT Salary
      FROM     EMPLOYEE;
```

Query 16. Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee.

```
Q16:  SELECT   E.Fname, E.Lname
      FROM     EMPLOYEE AS E
      WHERE    E.Ssn IN ( SELECT   Essn
                          FROM     DEPENDENT AS D
                          WHERE    E.Fname=D.Dependent_name
                                   AND E.Sex=D.Sex );
```

Query 18. Retrieve the names of all employees who do not have supervisors.

```
Q18:  SELECT   Fname, Lname
      FROM     EMPLOYEE
      WHERE    Super_ssn IS NULL;
```

Query 20. Find the sum of the salaries of all employees of the 'Research' department, as well as the maximum salary, the minimum salary, and the average salary in this department.

```
Q20:  SELECT   SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
      FROM     (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
      WHERE    Dname='Research';
```

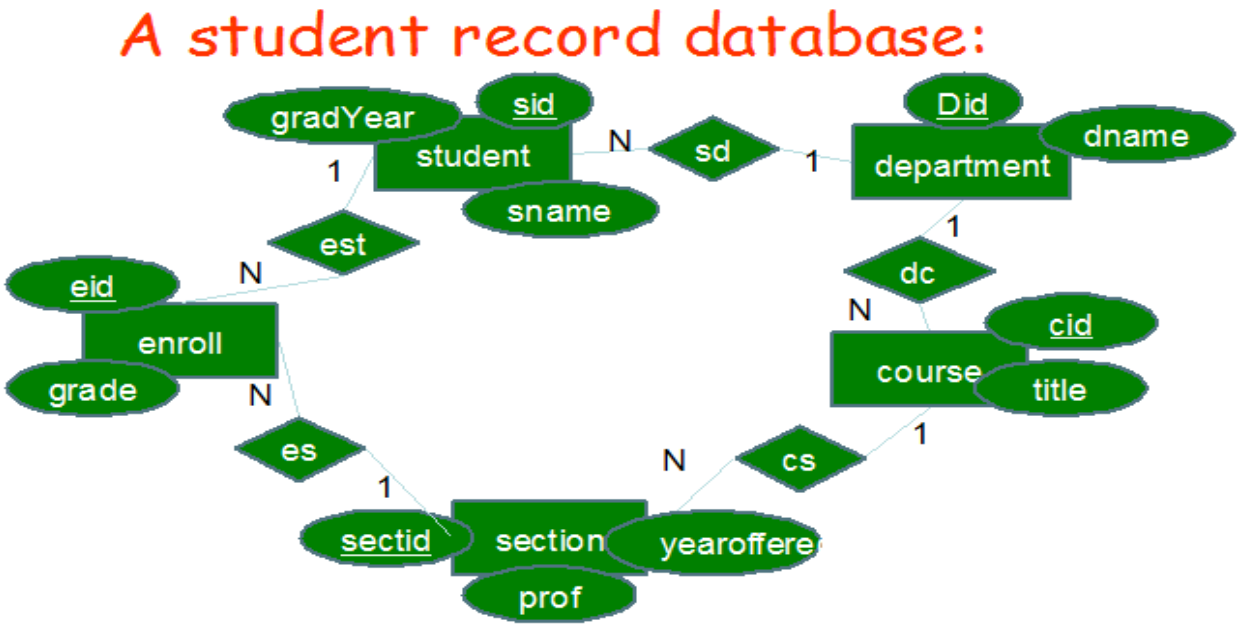
Queries 21 and 22. Retrieve the total number of employees in the company (Q21) and the number of employees in the 'Research' department (Q22).

```
Q21:  SELECT   COUNT (*)
      FROM     EMPLOYEE;

Q22:  SELECT   COUNT (*)
      FROM     EMPLOYEE, DEPARTMENT
      WHERE    DNO=DNUMBER AND DNAME='Research';
```

Query 28. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than $40,000.

```
Q28:  SELECT   Dnumber, COUNT (*)
      FROM     DEPARTMENT, EMPLOYEE
      WHERE    Dnumber=Dno AND Salary>40000 AND
               ( SELECT       Dno
                 FROM         EMPLOYEE
                 GROUP BY Dno
                 HAVING       COUNT (*) > 5)
```

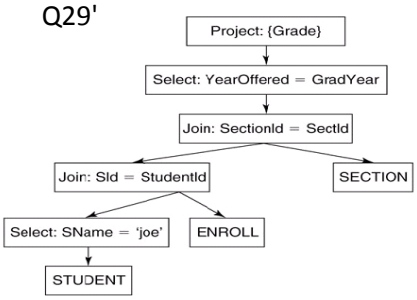
Examples on STUDENT DB :

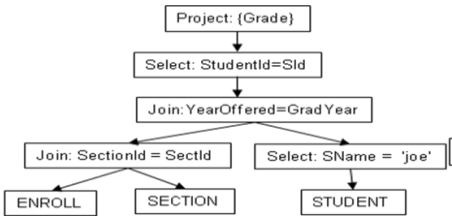


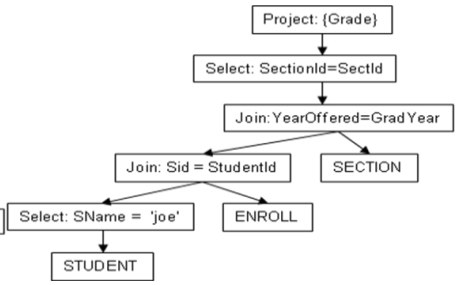

```
STUDENT(SId, SName, GradYear, MajorId)
DEPT(DId, DName)
COURSE(CId, Title, DeptId)
SECTION(SectId, CourseId, Prof, YearOffered)
ENROLL(EId, StudentId, SectionId, Grade)
```

1.) "Find the grades Joe received during his graduation year." circular sorgusu için 1 adet sorgu ağacı verilmiştir. Aynı sorgu için çok sayıda farklı sorgu (ağacı) yazılabilir. Bu ağaçlardan 2 tanesini çiziniz.

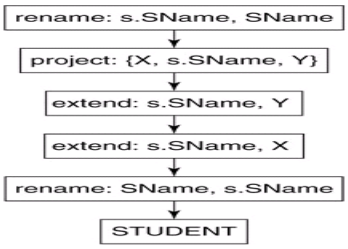
Q29'







2) select s.SName AS X, s.Sname, s.Sname AS Y
from STUDENT s
Sorgusuna ait sorgu ağacını çiziniz..



3.) T ilşkisel tablosu {A,B,C}özelliklerine sahiptir. Buna gore aşağıdaki operatörün dengini extend ve project operatörlerini kullanarak yazınız.

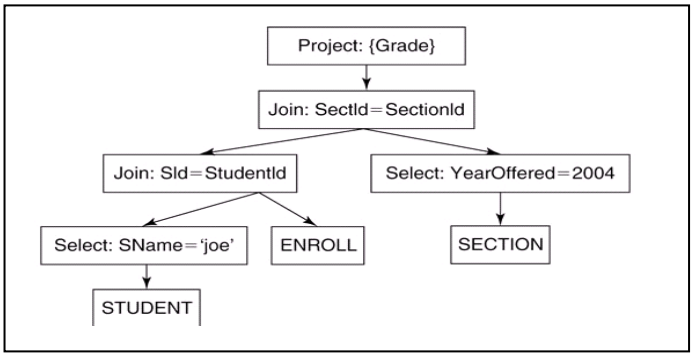
rename(T, C, D) ≡ ? $\pi_{A,B,D}(E_{C,D}(T))$

4.) Sunumlardaki Q22: "STUDENT ve DEPT kayıtlarının bütün kombinasyonlarını göster" sorgusu sadece join kullanarak nasıl yazılabilir?

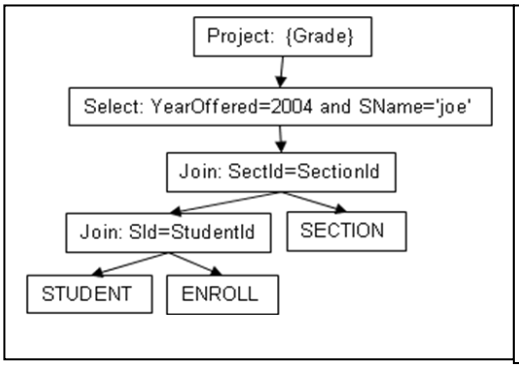
$(STUDENT \bowtie_{SId=SId} DEPT)$

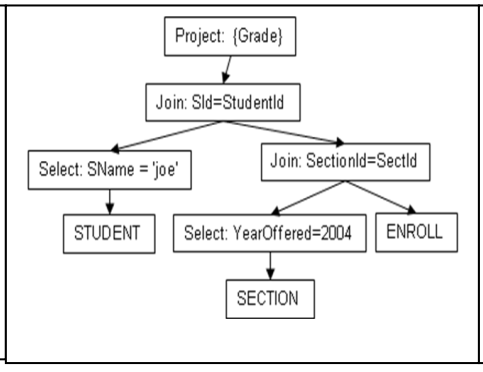
5.) Q22: product (STUDENT,DEPT) sorgusunun sonucu <u>eğer DEPT tablosu boş ise</u> ne olur?
Boş tablo olur.

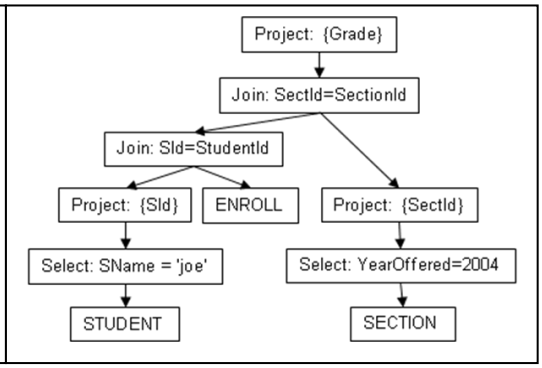
6.)



Yukarıdaki sorgu ağacına eşdeğer 3 farklı sorgu ağaci çiziniz..







7.) Sunumlardaki Q34'ye eşdeğer olan SQL sorgusunu yazınız.."List students that have the same major as Joe."

*select s2.**
from STUDENT s1, STUDENT s2
where s1.MajorId=s2.MajorId and s1.SName='joe'

8.) (T1 $\ltimes_{A<>B}$ T2) işleminin neticesi için ne söylenebilir?

Yüksek ihtiamalle T1 in kendisini verecektir. Bununla beraber T1'de herhengi bir r kaydının çıktıda gözükmemesi için; bu r'ye ait T1.A değerinin bütün T2.B'de aynı olması gerek! Anacak bu durumda A<>B sağlanmıyor ve bu r kaydı sonuca yansımıyor.

9.) (T1 $\ltimes_{A<>B}$ T2) ≡ (T1 $\triangleright_{A=B}$ T2) denkliği için ne söylenebilir?

Denk değildir. İlkinde en az 1 T1.A ,T2.B'ye eşit olmaması yeterli. Fakat ikincisinde T1.A, T2.B'ye hiç eşleşmeyecek. (T1-T2)

SId	SName	GradYear	MajorId
1	joe	2004	10
2	amy	2004	20
3	max	2005	10
4	sue	2005	20
5	bob	2003	30
6	kim	2001	20
7	art	2004	30
8	pat	2001	20
9	lee	2004	10

10.)

DEPT	DId	DName
	10	compsci
	20	math
	30	drama

Olduğuna göre; Aşağıdaki SQL veya ilişkisel cebir ifadelerinin sounuclarını bulunuz.

- select s2.SId from STUDENT s1, STUDENT s2 where s1.SName="sue'' AND s1.MajorId <> s2.MajorId
  a.) <1,3,5,7,9> b.) <1,4,6> c.) <2,6,8 > d.) <2,4> e.) < >
- select SId from STUDENT where GradYear = (select MAX(GradYear) from STUDENT);
  a.) <3> ***b.) <3,4>*** c.) < > d.) <6> e.) <2>
- select s1.SId from STUDENT s1 where NOT EXIST (select * from STUDENT s2 where s2.GradYear <s1.GradYear);
  a.) <6> b.) <3,4> c.) <> ***d.) <6,8>*** e.) bütün STUDENT tablosu
- select MajorId from STUDENT where GradYear>2003 group by MajorId having count(SId) >2;
  a.) <10> b.) <> c.) <10,20> d.) <30> e.) <10,10>
- select DName
  from DEPT d
  where NOT EXIST (select *
  from STUDENT s1, STUDENT s2
  where d.DId=s1.MajorId AND s1.MajorId=s2.MajorId AND s1.GradYear=s2.GradYear)
  a.) <compsci> b.) <math> ***c.) <drama>*** d.) < compsci ,drama> **e.) <>**
- semijoin(DEPT, STUDENT, DId <> MajorId) ifadesinin sonucu ne olur?
  a.) <10 compsci> b.) <20 math> c.) <30 drama> ***d.) < 10 compsci , 20 math, 30 drama>*** e.) <>
- antijoin(DEPT, STUDENT, DId = MajorId) ifadesinin sonucu ne olur?
  a.) <10 compsci> b.) <20 math> c.) <30 drama> d.) < 10 compsci , 20 math, 30 drama> ***e.) <>***

11.) SQL cümlesinin from kısmındaki tabla isimlerinin sırasının bir önemi var mıdır? Nedeni ile beraber yazınız.

Yoktur. Çünkü kartezyen çarpımda değişme özelliği vardır. AxB = BxA

12.) Aşağıdaki Q84 bir **group by** sorgusu mudur?

Q84: select max (q.numAs) AS maxAs from Q83 q

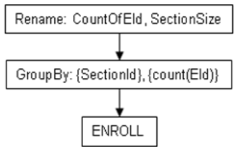
Evet.. Aggragate fonksiyonlar ancak group by altında çalışır. Fakat bu (ve benzeri sorgularda) group by boş olduğu için yazılmıyor..

13.) Sunumlarda, Q98'i içiçe olmayan sorgu kullanarak yazınız.. "Determine those students who took a course with Prof. Einstein."

```
Q98a =   select s.*
         from STUDENT s, ENROLL e, SECTION k
         where s.SId=e.StudentId and e.SectionId=k.SectId
         and k.Prof = 'einstein'
```

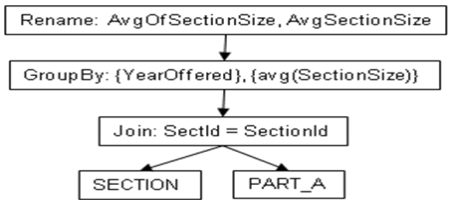
14.) Aşağıdaki sorgulara ait RA ifadesi ve sorgu ağacını ve SQL ifadesi, yazınız.

a) The number of students in each section. / *her ders-grubunun öğrenci sayısı*



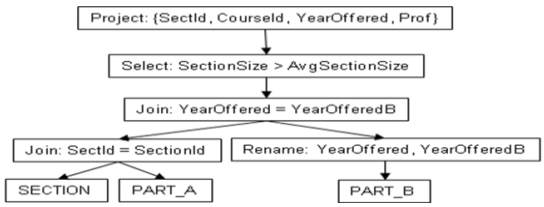

```
select e.SectionId, count(e.EId) as
SectionSize
from ENROLL e
group by e.SectionId
```

b) The average section size per year taught. */her sene ortalama ders-grub büyüklüğü*



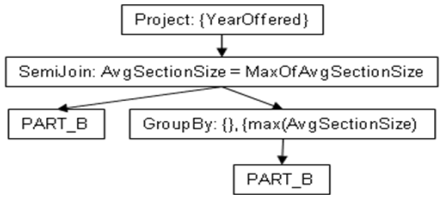

```
// Here, PART_A is the query from part (a) of this exercise
select k.YearOffered, avg(pa.SectionSize) as AvgSectionSize
from SECTION k, PART_A pa
where k.SectId=pa.SectionId
group by k.YearOffered
```

c) The sections that were larger than the average section size for their year. / *Açıldığı senedeki ortalama ders-grub yüyüklüğünden fazla olan ders-grubları*



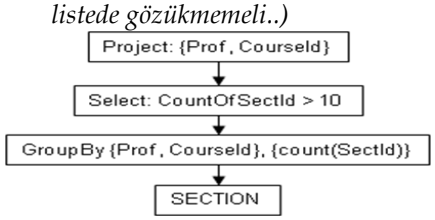

```
select k.*
from SECTION k, PART_A pa, PART_B pb
where k.SectId=pa.SectId and k.YearOffered=pb.YearOffered
      and pa.SectionSize > pb.AvgSectionSize
```

d) The year having the largest average section size. / *ortalama ders-grubu büyüklüğünün en fazla olduğu sene*



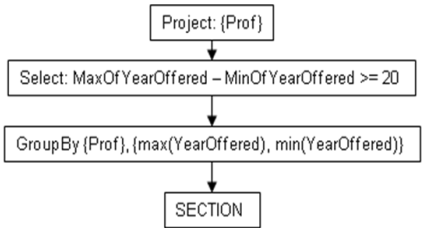

```
select pb.YearOffered
from PART_B pb
where pb.AvgSectionSize in (select max(p.AvgSectionSize)
                            from PART_B p)
```

e) The (professor, course) pairs such that the professor taught that course more than 10 times. */profesorün ismi ve (eğer varsa) 10 defadan fazla vermiş oldukları dersin ismi. (eger bu koşulu sağlayan ders vermediyse profesorün ismi de listede gözükmemeli..)*



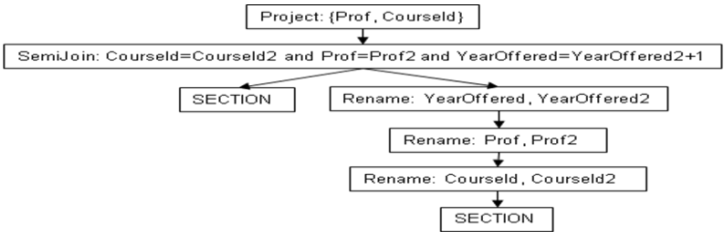

```
select k.Prof, k.CourseId
from SECTION k
group by k.Prof, k.CourseId
having count(k.SectId)>10
```

f) The professors who have taught courses over a range of at least 20 years. (That is, the latest course taught comes at least 20 years after the earliest course taught.) / *En az 20 senelik hocalık yapan profesörlerin ismi*



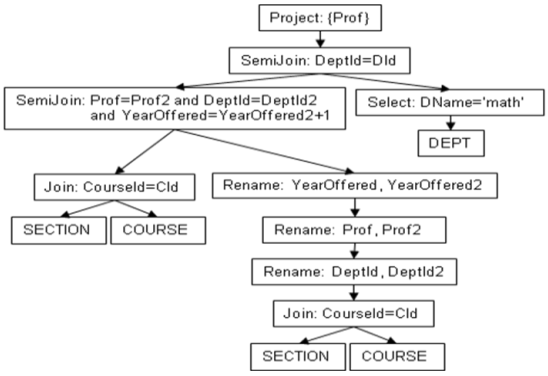

```
select k.Prof
from SECTION k
group by k.Prof
having max(YearOffered) – min(YearOffered) >= 20
```

g) The (professor, course) pairs such that the professor taught that course for at least two consecutive years. / *Ard arda en az 2 sene üstüste aynı derse girmiş olan profesorun ismi ve bu dersin id'si..*



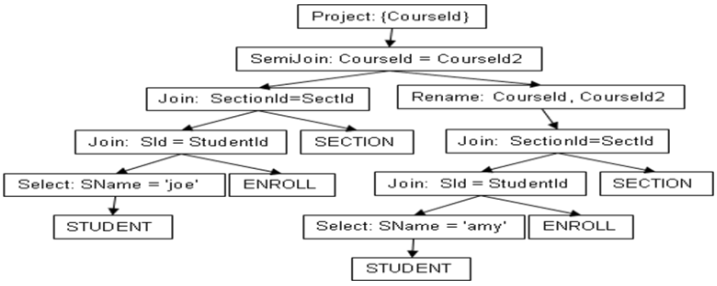

```
select k1.Prof, k1.CourseId
from SECTION k1, SECTION k2
where k1.CourseId=k2.CourseId
and k1.Prof = k2.Prof
and k1.YearOffered=k2.YearOffered+1
```

h) The professors who taught math courses for at least two consecutive years. / *Ard arda en az 2 sene üstüste math bölümüne ait derse girmiş profesörün isimlerinin listesi..*



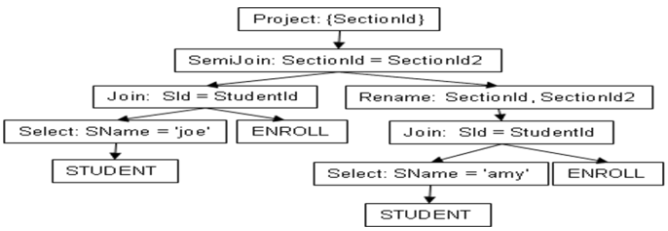

```
select k1.Prof
from SECTION k1, SECTION k2, COURSE c1, COURSE c2, DEPT d
where k1.CourseId=c1.CId and c1.DeptId = d.DId
and k2.CourseId=c2.CId and c2.DeptId = d.DId
and k1.Prof=k2.Prof
and k1.YearOffered=k2.YearOffered+1
and d.DName = 'math'
```

i) The courses that Joe and Amy have both taken. / *Joe ve Amy isimli öğrencilerin aldıkları ortak derslerin id'leri*



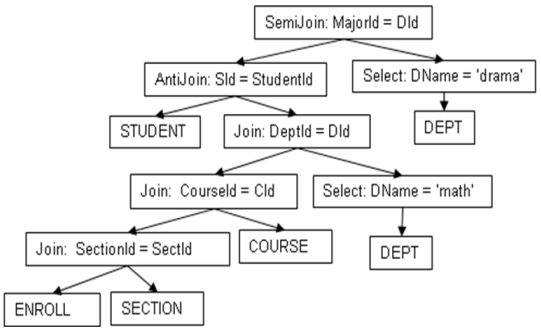

```
select k1.CourseId
from STUDENT s1, ENROLL e1, SECTION k1,
     STUDENT s2, ENROLL e2, SECTION k2
where s1.SId=e1.StudentId and
e1.SectionId=k1.SectId
and s2.SId=e2.StudentId and
e2.SectionId=k2.SectId
and s1.SName='joe' and s2.SName='amy'
and k1.CourseId=k2.CourseId
```

j) The sections that Joe and Amy took together. / *Joe ve Amy isimli öğrencilerin beraber aldıkları ders-gruplarının id'leri*



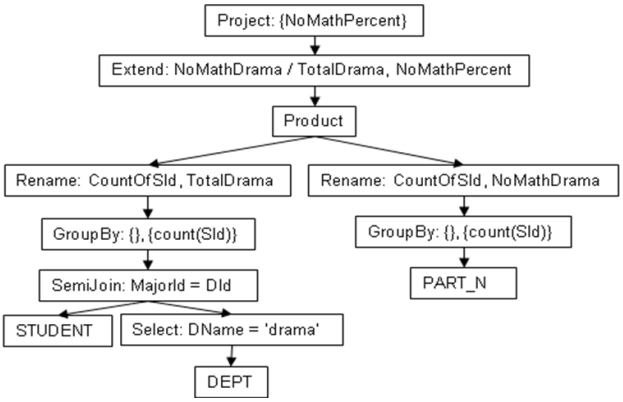

```
select e1.SectionId
from STUDENT s1, ENROLL e1, STUDENT s2,
ENROLL e2
where s1.SId=e1.StudentId and
s2.SId=e2.StudentId
and s1.SName='joe' and s2.SName='amy'
and e1.SectionId=e2.SectionId
```

k) The drama majors who never took a math course. / *Math bölümünden hiç ders almamış olan drama öğrencilerinin isimleri*



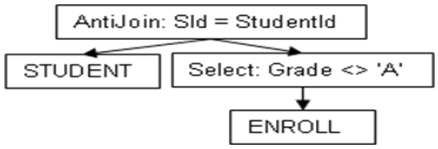

```
select s.*
from STUDENT s, DEPT d
where s.MajorId=d.DId and d.DName='drama'
and s.SId not in
      (select e.StudentId
       from ENROLL e, SECTION k, COURSE c, DEPT d2
        where e.SectionId=k.SectId
```

l) The percentage of drama majors who never took a math course. / *Math bölümünden hiç ders almamış olan drama öğrencilerinin toplam drama öğrencilerine oranı (yüzdesel olarak)*



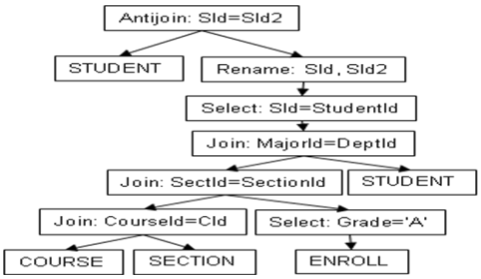

```
// Step 1:  Calcuate the total number of drama majors
select count(s.SId) as TotalDrama
from STUDENT s, DEPT d
where s.MajorId=d.DId and d.DName='drama'

// Step 2:  Calculate the number of drama majors from part (n)
select count(pn.SId) as NoMathDrama
from PART_k pk

Calculate the desired output
select s2.NoMathDrama / s1.TotalDrama as NoMathPercent
from STEP_1 s1, STEP_2 s2
```

m) The STUDENT record of the students who never received a grade below an "A". / *Bütün notlari 'A' olan öğrecilere ait bilgiler.*




```
select s.*
from STUDENT s
where s.SId not in (select e.StudentId
```

n) The STUDENT record of the students who never received a grade of "A" in any course they took in their major. / *Kendi bölümündeki hiç bir dersten*



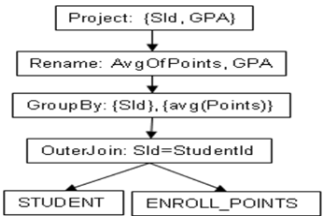

```
select s.*
from STUDENT s
where s.SId not in
        (select s2.SId
          from STUDENT s2, ENROLL e, SECTION k, COURSE c
         where s2.SId=e.StudentId  and e.SectionId=k.SectId
                                   and k.CourseId=c.CId
                                  and c.DeptId=s2.MajorId
                                    and e.Grade='A')
```

15.) Veri tabanında ENROLL tablosu her ders kaydına ait alınan notu harf olarak 'Grade' niteliğinde tutmaktadır. Öğrenci ortalamaları ile ilgili sorgular için; mevcut şemaya ek olarak GRADEPOINTS(LetterGrade, Points) isimli tablo da ekleniyor. Bu tabla her harf notun, numeric değerini vermektedir. {('A', 4.0) ('A-', 3.7) ('B+', 3.3) ….} gibi.. Buna gore aşağıdaki sorgulara ait RA ifadesi ve sorgu ağacını ve SQL ifadesi, yazınız.

(Not Aşağıda istenen sorgularda ENROLL yerine, ENROLL_POINTS (EId,StudentId, SectionId, Grade, Point) view tablosunu tanımlayıp, kullanmaniz cevapları kolaylaştırıyor…)

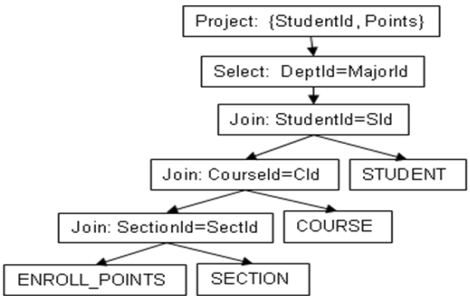
```
create view ENROLL_POINTS as
    select e.*, g.Points
    from ENROLL e, GRADEPOINTS g
    where e.Grade=g.LetterGrade
```

a) The GPA of each student. */ her öğrencinin not ortalaması (her öğrencinin: yani herhangi bir ders kaydı yaptırmamış olanlar da dahil)*

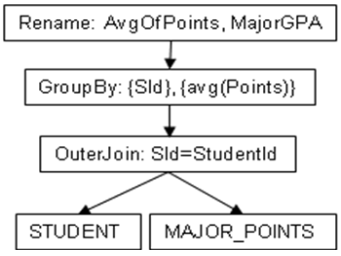



```
select s.SId, avg(ep.Points) as GPA
from STUDENT s outer join ENROLL_POINTS ep on
                                s.SId=ep.StudentId
group by s.SId
```

b) The "major GPA" of each student. */Her öğrencini Major GPA ortalaması. (yani bağlı olduğu bölümden aldığı derslerin ortalaması)*

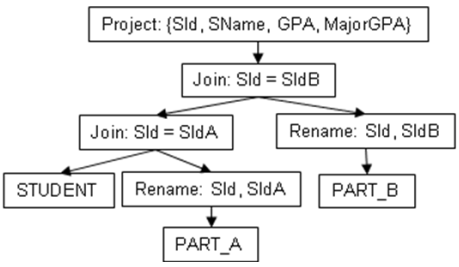



```
create view MAJOR_POINTS as
    select ep.StudentId, ep.Points
    from ENROLL_POINTS ep, SECTION k, COURSE c, STUDENT s
    where ep.SectionId=k.SectId and k.CourseId=c.CId
                                and c.DeptId=s.MajorId
```



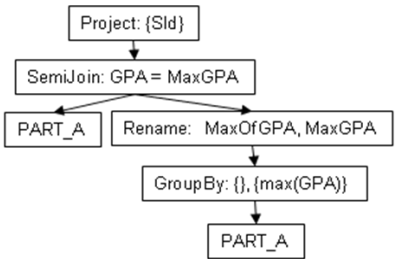

```
select s.SId, avg(mp.Points) as MajorGPA
from STUDENT s outer join MAJOR_POINTS mp on s.SId=mp.StudentId
group by s.SId
```

c) For each student, a record containing the student's ID, name, GPA, and major GPA. */ Her öğrencinin numarası, ismi, genel not ortalaması ve major not ortalaması (her öğrencinin: yani herhangi bir ders kaydı yaptırmamış olanlar da dahil)*



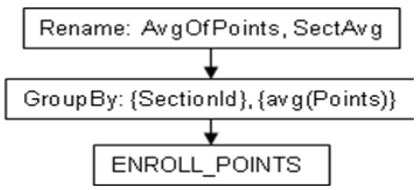

```
select s.SId, s.SName, pa.GPA, pb.MajorGPA
from STUDENT s, PART_A pa, PART_B pb
where s.SId=pa.SId and s.SId=pb.SId
```

d) The student(s) having the highest GPA */ En yüksek GPA değerine sahip olan öğrecilerin id'leri..*



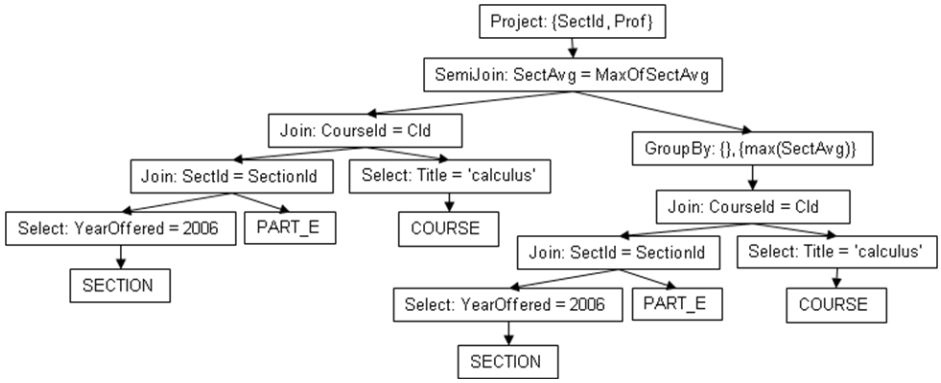

```
select pa.SId
from PART_A pa
where pa.GPA in (select max(pa2.GPA) from PART_A pa2)
```

e) The average grade points per section. */ Her ders-grubuna ait ortalama.*




```
select ep.SectionId, avg(gp.Points) as SectAvg
from ENROLL_POINTS ep
group by ep.SectionId
```

f) The section of calculus taught in 2006 having the highest average grade, and the professor who taught it. */ 2006'da verilen 'calculus' ders-grublarından sınıf ortalaması en yüksek olan grubun id'si ve hocasının ismi*



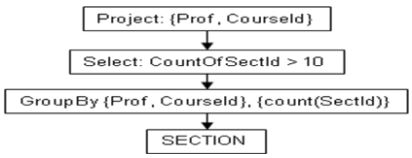

```
create view SECTS_2006 as
        select k.SectId, k.Prof, pe.SectAvg
    from SECTION k, PART_E pe, COURSE c
  where k.CourseId=c.CId and k.SectId=pe.SectionId
    and c.Title='calculus' and k.YearOffered=2006

select s1.SectId, s1.Prof
from SECTS_2006 s1
where s1.SectAvg in (select max(s2.SectAvg) from SECTS_2006 s2)
```

16.) DEAN Office için oluşturulan STUDENT_INFO (SId, SName,GPA, NumCoursesPassed, NumCoursesFailed) görüntüsünü (view) tanımlayınız...

```
// Buradaki ENROLLPOINTS, onceki soruda tanımlanan notların sayısal değerini içeren tablo.
create view GPA as
    select s.SId, avg(g.Points) as GPA
    fromSTUDENT s outer join ENROLL e on s.SId=e.StudentId,ENROLLPOINTS g
    where e.Grade=g.Grade
    group by s.SId

create view NUMPASSED as
    select s.SId, count(e.Eid) as NumCoursesPassed
    from STUDENT s outer join ENROLL e on s.SId=e.StudentId
    where e.Grade <> 'F'
    group by s.SId
```

```
create view NUMFAILED as
    select s.SId, count(e.Eid) as NumCoursesFailed
    from STUDENT s outer join ENROLL e on s.SId=e.StudentId
    where e.Grade = 'F'
    group by s.SId

create view STUDENT_INFO as
    select s.SId, s.SName, g.GPA,
       p.NumCoursesPassed, f.NumCoursesFailed
    from STUDENT s, GPA g, NUMPASSED p, NUMFAILED f
    where s.SId=g.SId and s.SId=p.SId and s.SId=f.SId
```

17.) Aşağıdaki ilişkisel cebir (ağacı) veya SQL cümleleri hangi işlemi yaptığını bulunuz..



a) 10'dan fazla ders veren Profesorlerin isimleri ve verdikleri dersin id'sini buluyor.
b) Aynı dersi 10'dan fazla kez vermiş olan profesorlerin isimleri ve verdikleri dersin id'sini buluyor.
c) 10'dan fazla section'ı olan dersler için, dersi veren profesor ismi ve dersin id'sini veriyor.
d) 10 farklı profesor tarafından verilmiş olan dersler için profesorlerin isimleri ve dersin id'sini veriyor.
e) hiçbiri

```
Q1: select (ENROLL, Grade = 'F')
Q2: antijoin (SECTION,Q1, SectId = SectionId)
Q3:rename (Q2, Prof, selectedProf)
Q4: antijoin (SECTION, Q3, Prof = selectedProf)
Q5: groupby (Q4,{Prof},{})
```

a) Hiç F vermemiş olan profesorlerin isimlerini tekrarsız listeler.
b) Verdiği derslerin en az birinde en az 1 F vermiş olan profesörlerin isimlerini tekrarsız listeler.
c) Verdiği bütün derslerde En az 1 F vermiş olan profesörlerin isimlerini tekrarsız listeler.
d) Bazı derslerde F vermemiş olan profesörlerin isimlerini tekrarsız listeler.
e) Bazı derslerde F vermiş olan profesörlerin isimlerini tekrarsız listeler.

```
select s.*
from STUDENT s
where s.SId not in (select e.StudentId
                    from ENROLL e
                    where e.Grade <> 'A');
```

a) Hiç A almamış öğrencilerin niteliklerini listeler.
b) En az 1 tane A almış öğrencilerin niteliklerini listeler.
c) A'dan başka not almamış öğrencilerin niteliklerini listeler.
d) Hiç A verilmemiş section'lardaki öğrencilerin niteliklerini listeler.
e) Bazı derslerden A alamamış olan öğrencilerin isimlerini listeler.

```
select s.*
from STUDENT s, DEPT d
where s.MajorId=d.DId and d.DName='drama'
and s.SId not in
        (select e.StudentId
                from ENROLL e, SECTION k, COURSE c, DEPT d2
                where e.SectionId=k.SectId and k.CourseId=c.CId and
                      c.DeptId=d2.DId and d2.DName='math')
```

a) 'math' bölümünden hiç ders almamış olan drama bölümü öğrencilerini listeler.
b) Matematik ve drama bölümü öğrencisi olup math bölümünden ders almamış olan öğrencileri listeler.
c) Bölümü (major'ı) sadece drama olan öğrencileri listeler.
d) Bölümü (major'ı) sadece math olan öğrencileri listeler.
e) drama ve math Bölümü (major'ı) dışındaki öğrencileri listeler.

```
select k1.Prof
from SECTION k1, SECTION k2, COURSE c1, COURSE c2, DEPT d
where k1.CourseId=c1.CId and c1.DeptId = d.DId and
      k2.CourseId=c2.CId and c2.DeptId = d.DId and k1.Prof=k2.Prof
      and k1.YearOffered=k2.YearOffered+1 and d.DName = 'math' ;
```

a) Matematik bölümündeki profesörlerden 'math' dersine en az 2 kez girmiş olanların ismileri
b) En fazla 2 kez 'math' bölümünde derse girmiş olan profesörlerin isimleri
c) Tam iki sene üstüste 'math' bölümünde ders vermiş olan profesörlerin isimleri
d) En az 2 sene ardarda 'math' bölümünde ders vermiş olan profesörlerin isimleri
e) 'math' bölümünde en az 2 senedir çalışan profesörlerin isimleri

18-) SQL ile ilşkisel cebir (relational algebra) hakkında aşağıdakilerden hangisi doğrudur?

a.) SQL nasıl-odaklı iken, ilşkisel cebir sonuç-odaklıdır.
b.) İlişkisel cebir ticari veri tabanlarında standard kullanıcı sorgulama dilidir.
c.) SQL sorgusunun çalıştırılması için ilk olarak ilşkisel cebir ifadesine dönüştürülür.
d.) İlişkisel cebir veri tanımlama dili (*data definition lang.*) olanaklarını içerir.

19-)

Emp(*eid:* integer, *ename:* string, *age:* integer, *salary:* real)
Works(*eid:* integer, *did:* integer, *pct time:* integer) Works.*eid,* Works.*did* yabancı anahtar
Dept(*did:* integer, *budget:* real, *managerid:* integer) Dept.*managerid* yabancı anahtar

A.) Yukarıdaki veri tabanında oluşturulan aşağıdaki View'lardan hangisinde veya hangilerinde olabilecek bir yenilik ana tabloda (tablolarda) "otomatik yanilenmeye" olanak sağlar? *(Doğru cevabı/cevaplara ait numarayı yuvarlak içine alınız)*

1) CREATE VIEW SeniorEmp (name, age, salary)
AS SELECT E.ename, E.age, E.salary
FROM Emp E
WHERE E.age > 50

2.) CREATE VIEW SeniorEmp (eid, name, age, salary)
AS SELECT E.eid, E.ename, E.age, E.salary
FROM Emp E
WHERE E.age > 50

3.) CREATE VIEW AvgSalaryByAge (age, avgSalary)
AS SELECT E.eid, AVG (E.salary)
FROM Emp E
GROUP BY E.age

B.) Bu şirkette 30 yaşından daha genç yönetici olmasını istemiyoruz. Bununla ilgili olarak iki farklı yaklaşım aşağıda gösterilmiştir. Birincisi bunu Dept tablosunda "tablo kısıtlaması" olarak belirtmek, diğeri ise ASSERTION yazmak.

1.yol:
```
CREATE TABLE Dept (
    did INTEGER,
    buget REAL,
    managerid INTEGER ,
    PRIMARY KEY (did),
    FOREIGN KEY (managerid) REFERENCES Emp,
     CHECK( (SELECT E.age FROM Emp E, Dept D)
              WHERE E.eid = D.managerid ) > 30 )
```

2.yol:
```
CREATE ASSERTION managerAge
CHECK (NOT EXIST(SELECT *
                 FROM Emp E, Dept D
                 WHERE E.eid = D.managerid
                       AND E.age<30 ) )
```

Bu iki yoldan hangisi daha doğrudur? Neden?

ikinci yol daha esnek. Çünkü kısıtlamada sonradan değişiklik yapmak mümkün. Birinci yolda ise tabloyu silip tekrar oluşturmak gerekiyor. Birinci yoldaki kisitlar sadece INSERT TABLE komutlarında etkin olur.

C.) Aşağıdaki sorgu ne yapar?

```
SELECT did  FROM Dept D
WHERE NOT EXIST
      (SELECT *
       FROM Emp E1, Emp E2, Works W1, WORKS W2
       WHERE E1.eid=W1.eid AND E2.eid=W2.eid
             AND W1.did=W2.did AND W1.did=D.did
            AND E1.salary=E2.salary)
```

Aynı maaşlı elemanların bulunmadığı, yani bütün çalışanların maaşlarının farklı olduğu bölümleri bulur.

20-) **Suppliers**(*sid:* integer, *sname:* string, *address:* string)
Parts(*pid:* integer, *pname:* string, *color:* string)
Catalog(*sid:* integer, *pid:* integer, *cost:* real) Catalog.*sid*, Catalog.*pid* yabancı anahtarlar

Yukarıdaki veri tabanı üzerinde yazılmış olan aşağıdaki ilşkisel algebra ve SQL ifadelerini birbiri ile aynı işi yapacak şekilde eşleştirin.

$\pi_{sname}(\pi_{sid}((\pi_{pid}\sigma_{color='red'}Parts) * Catalog * Suppliers)$	C
$\rho(R1, \pi_{sid}((\pi_{pid}\sigma_{color='red'}Parts) * Catalog))$ $\rho(R2, \pi_{sid}((\pi_{pid}\sigma_{color='green'}Parts) * Catalog))$ $R1 \cap R2$	D
$(\pi_{sid,pid}Catalog) \div (\pi_{pid}\sigma_{color='red' \text{ OR } color='green'}Parts)$	E
$\pi_{sid}(\pi_{pid}(\sigma_{color='red' \text{ } color='green'}Parts) * Catalog)$	A
$(\pi_{sid,pid}Catalog) \div (\pi_{pid}\sigma_{color='red'}Parts)$	B

```
SELECT C.sid
FROM Catalog C, Parts P
WHERE (P.color = 'red' OR P.color = 'green')
        AND P.pid = C.pid
```

```
SELECT C.sid
FROM Catalog C
WHERE NOT EXISTS (SELECT P.pid
                  FROM Parts P
                  WHERE P.color = 'red'
                  AND (NOT EXISTS (SELECT C1.sid
                                   FROM Catalog C1
                                   WHERE C1.sid = C.sid AND
                                         C1.pid = P.pid)))
```

```
SELECT S.sname
FROM Suppliers S, Parts P, Catalog C
WHERE P.color='red' AND C.pid=P.pid AND C.sid=S.sid
```

```
SELECT C.sid
FROM Parts P, Catalog C
WHERE P.color = 'red' AND P.pid = C.pid
AND EXISTS (SELECT P2.pid
            FROM Parts P2, Catalog C2
            WHERE P2.color = 'green' AND C2.sid = C.sid
                  AND P2.pid = C2.pid )
```

```
SELECT C.sid
FROM Catalog C
WHERE NOT EXISTS (SELECT P.pid
                  FROM Parts P
                  WHERE (P.color = 'red' OR P.color = 'green')
                        AND (NOT EXISTS(SELECT C1.sid
                                        FROM Catalog C1
                                        WHERE C1.sid = C.sid
                                        AND C1.pid = P.pid)))
```

21.) Öğrenci (isim, no, gpa, danışman_ismi) *NOT: danışman_ismi, Öğretmen'e yabancı anahtar*
Öğretmen (isim, bölüm_ismi,maaş)

Bir **Öğrenci-Öğretmen** veri tabanına ait iki tablo üzerinde aşağıdaki sorguların **<u>hangisi (hangileri)</u>**

A.) ""Bilgisayar" isimli bölümdeki öğrencilerin ismini" verir? (*cevab(lar)ın numarasını yuvarlak içine alınız..)*

```
SELECT   Öğrenci.isim
FROM     Öğrenci
WHERE  Öğrenci.danışman_ismi =(SELECT Öğretmen.isim FROM Öğretmen WHERE
Öğretmen.bölüm_ismi= "Bilgisayar");
```

```
SELECT   Öğrenci.isim
FROM     Öğrenci, Öğretmen
WHERE  Öğrenci.danışman_ismi =Öğretmen.isim AND Öğretmen.bölüm_ismi= "Bilgisayar";
```

```
SELECT   Öğrenci.isim
FROM     Öğrenci
WHERE  Öğrenci.danışman_ismi  IN (SELECT Öğretmen.isim FROM Öğretmen WHERE
Öğretmen.bölüm_ismi= "Bilgisayar");
```

B.) "Bütün öğrencilerinin Gpa (ortalaması)'inin ortalaması 3,5 üzeri olan öğretmenlerin ismini" verir? (*cevab(lar)ın numarasını yuvarlak içine alınız..)*

```
SELECT   Öğrenci.danışman_ismi, AVG(Öğrenci.gpa)
FROM     Öğrenci
WHERE AVG(Öğrenci.gpa)>3.5
GROUP BY Öğrenci.danışman_ismi;
```

```
SELECT   Öğrenci.danışman_ismi, AVG(Öğrenci.gpa)
FROM     Öğrenci
GROUP BY Öğrenci.danışman_ismi
HAVING AVG(Öğrenci.gpa>3.5;
```

22.)

Yandaki R, S1, S2 ve S3 tablolari için aşağıdaki SQL sorgu sonuçlarını bulunuz. (Not: R tablosundaki X, Y ve Z sırasıyla S1(X), S2(Y) ve S3(Z) ye yabanci anahtardır(foreign key))

(cevaplarda tekrar sayısı önemli!)

R

X	Y	Z	b
1	k	1	200
1	l	1	300
2	m	1	100
3	k	2	500
3	m	2	400
3	l	2	200
4	l	2	1000
4	m	3	100
5	l	3	600
5	m	3	200
5	k	3	200
5	l	3	200

S1

X	t
1	a
2	b
3	a
4	b
5	a
6	b

S2

Y
k
l
m

S3

Z
1
2
3

```
SELECT t
FROM S1
WHERE EXISTS
(select * from R where R.y='m' AND S1.x=R.x);
```

b,a,b,a

```
SELECT R.z
FROM R
GROUP BY R.z
HAVİNG count(R.z)>3
```

2
3

```
SELECT S1.t, count(S1.t)
FROM S1
GROUP BY S1.t
```

a 3
b 3

```
SELECT R1.x
FROM R R1,R R2
WHERE R1.x=R2.z
```

1,1,1,1,1,1,2,2,2,2, 3,3,3,3,3,3,3,3,3,3,3,3,3,3

```
SELECT Temp.RRZ, Temp.min
FROM (SELECT RR.Z as RRZ, RR.Y, min(RR.b) as min
        FROM R RR
        GROUP BY RR.Z, RR.Y ) AS Temp
WHERE Temp.min = (SELECT min(Temp.min) FROM Temp);
```

1 100
3 100

```
SELECT R.Z
FROM R
WHERE b > 200
GROUP BY R.Z
HAVING count(*)>2;
```

2

```
(SELECT x FROM S1)
MINUS
(SELECT x FROM R)   sorgusu sonucu 6 olduğuna göre aşağıdaki sorgu sonucu ne olur?

SELECT x FROM S1
WHERE NOT EXIST ((SELECT y FROM S2)
                 MINUS
                 (SELECT R.y
                 FROM R
                 WHERE R.x=S1.x))
```

3
5

23) **Student**(snum *integer*, sname *string*, major *string*, level *string*, age *integer*)
Class(name *string*, meets_at *string*, room *string*, fid *integer*)
Enrolled(snum *integer*, cname *string*) *(Not: snum, Student'a yabancı anahtar, cname Class'a yabancı anahtar)*

Bu veri tabanında Öğrenci özellikleri **Student** tablosunda, Ders özellikleri **Class** tablosunda saklanırken; her ğrencinin aldığı dersler ise **Enrolled** tablsosunda tutulmaktadır. Buna göre aşağıdaki SQL ifadelerinin hangi sorguya cevap olacağını veya (trigger ise) ne yaptığını turkçe veya ingilizce olarak yazınız. *(Örnek cevap: "Bu Sorgu en küçük yaşa sahip öğrenciyi bulur"gibi bir ifade olmalı...)*

```
SELECT DISTINCT S.sname
FROM Student S
WHERE S.snum IN (SELECT E.snum
                 FROM Enrolled E
                 GROUP BY E.snum
                HAVING COUNT (*) >= ALL (SELECT COUNT (*)
                                         FROM Enrolled E2
                                         GROUP BY E2.snum ))
```

En fazla derse kayıtlı olan öğrencilerin isimlerini bulur.

```
SELECT DISTINCT S.sname
FROM Student S
WHERE S.snum NOT IN (SELECT E.snum
                     FROM Enrolled E )
```

Hiçbir derse kayıt yaptımamış olan öğrencilerin isimlerini bulur.

Student.sname niteliğini değiştirilemez hale getiren bir triggerdir.

```
CREATE TRİGGER aSimpleTrigger
AFTER UPDATE OF sname in Student
FOR EACH ROW
referencing old row as oldrow, new row as newrow
when oldrow.sname <> newrow.sname
set newrow.sname = oldrow.sname
```

24.) Aşağıda "Sipariş takip veri tabanına" ait ilişkisel model tabloları görülmektedir.

MÜŞTERİ (mno , isim, şehir)

SİPARİŞ (sno, starih, müşteri_no, toplam) *// toplam TL değeri, starih siparişin verildiği tarih*

SİPARİŞ_PARÇA (siparişno,parçano, miktar)

PARÇA(pno, birim fiyatı)

POSTA (sno, dno, ptarih) *// ptarih, siparişte istenenlerin depodan kargoya verildiği tarihtir.*

DEPO(dno, şehir)

a.) Yabancı anahtarları bulunuz..

Yabancı anahtarlar ***yana yatık ve bold*** olarak gosterilmiştir.

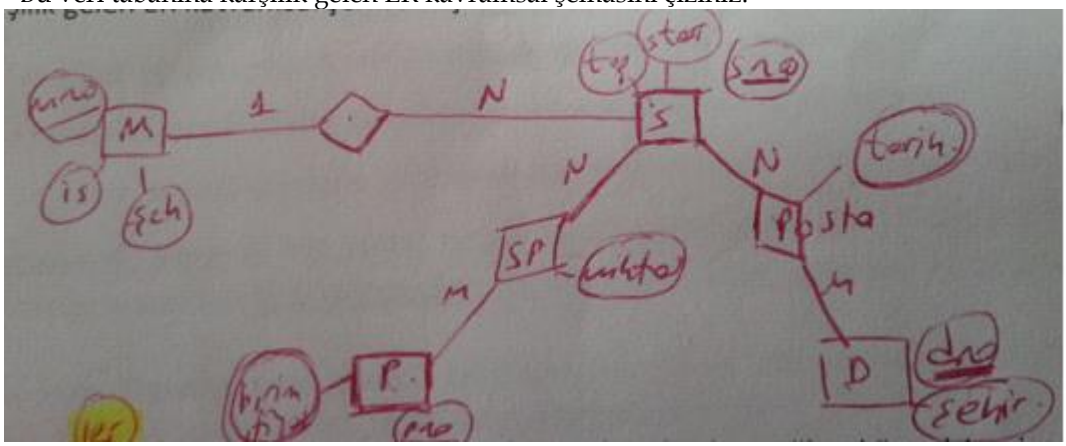
MÜŞTERİ (mno , isim, şehir)

SİPARİŞ (sno, starih, ***müşteri_no***, toplam)
SİPARİŞ_PARÇA (***siparişno,parçano,*** miktar)

PARÇA(pno, birim fiyatı)

POSTA (***sno, dno***, ptarih)

b.) Bu veri tabanına karşılık gelen ER kavramsal şemasını çiziniz.



c.) 'Ömer Aydın' isimli müşteriler tarafından verilen siparişlerin hangi depolardan sağlandığını, (siparişno, dno) olarak listeleyen **ilişkisel algebrayı** yazınız.

$$S \leftarrow \pi_{SNO}\ (\ SİPARİŞ \bowtie_{MÜŞTERİ\_NO=MNO} \sigma_{İSİM='ÖMER\ AYDIN'}(MÜŞTERİ)\ )$$

$$R \leftarrow \pi_{sno,dno}(POSTA*S)$$

d.)

$R1 \leftarrow \sigma_{ptarihi \leq starihi+30}(SİPARİŞ * POSTA)$
$RESULT \leftarrow \Pi_{sno}(SİPARİŞ) - \Pi_{sno}(R1)$

Yukarıdaki ilişkisel algebra hangi sorguya cevap verir?
.....

sipariş tarihinden itibaren 30 gün içinde kargoya verilmeyen siparişlerin nosu'nu verir

$R1 \leftarrow \sigma_{ptarihi > starihi+30}(SİPARİŞ * POSTA)$
$RESULT \leftarrow \Pi_{sno}(R1)$

e.) Yukarıdaki algebra d şıkkı ile aynı sorguya mı cevap verir? Fark varsa nedir?
.....

Fark var. Sipariş tarihinden 30 gün sonra postaya verilen siparişlerin no'sunu verir. .

25.) T1 (A,C) ve T2(B) tabloları için aşağıdaki tabloda yazılan ilişkisel algebra ve SQL ifadelerinden birbirine denk olanları belirleyin.

(ifadeler arasında dengi olmayan olabileceği gibi birden çok dengi olan da olabilir!…cevabınızın örneğin $Q21 \equiv Q22 \equiv Q23$ *ve* $Q30 \equiv Q31$ *gibi bir şekilde olmalı. Herhangi bir açıklama yapmayınız…)*

Q1: semijoin(T1, T2, A<>B)	Q2: rename(T1, C, D)	Q3: antijoin (T1, T2, A=B)	Q4: join(T1,T2,A=B)
Q5: outerjoin(T1,T2,A=B)	Q6: project(extend(T1, C, D), {A,D})	Q7: select(product(T1,T2),A=B)	Q8: Groupby(join(T1,T2,A<>B), {A,C},{ })
Q9: select * from T1 where A **IN** (select B from T2)	Q10: select * from T1 where A **NOT IN** (select B from T2)		

Denklik ifadeleri: ……***Q1 = Q8… … … … … … … … …***

… … … … … Q2 = Q6

……Q3=Q10

……Q4=Q7… … … … … ……

T1 (A,C) ve T2(B) tabloları için yazılan aşağıdaki Q1 ve Q2 sorguları için aşağıdakilerden hangisi söylenebilir?

Q1= semijoin (T1,T2, A=B)

Q2= antijoin (T1,T2, A=B)

a) Union(Q1,Q2) =T1
b) Union(Q1,Q2) =T2
c) Union(Q1,Q2) = boş tablo
d) Q1 tablosunun satır sayısı (*cardinality*) > Q2 tablosunun satır sayısı (*cardinality*)
e) Union(Q1,Q2) = join(T1,T2, a=b)

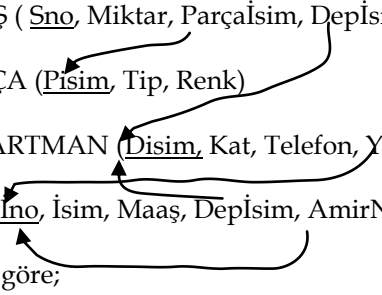
26.) Çok katlı bir alış-veriş merkezindeki satış bilgilerinin takip eden sistemin veri tabanı tabloları şu şekildedir: (Modelde özel anahtarlar ve yabancı anahtarlar gösterilmektedir)

SATIŞ (Sno, Miktar, ParçaİsimDepİsim)

PARÇA (Pisim, Tip, Renk)

DEPARTMAN (Disim, Kat, Telefon, YöneticiNo)

İŞÇİ (İno, İsim, Maaş, DepİsimAmirNo)



Buna göre;

```
SELECT DISTINCT S.Parçaisim
FROM SATIŞ S, DEPARTMAN D
WHERE S.DepİsimD.Disim AND Kat=2;
```

cümlesi hangi sorguya cevap verir. ***2. katta satılan parçaların isimlerini veriyor.***

Aynı işi yapan 3 farklı sorgu cümlesindeki boşlukları tamamlayın.

1.sorgu) ***IN kullanarak*** *SELECT DISTINCT S.Parçaisim* *FROM SATIŞ S* *WHERE S.DepİsimIN* ***(SELECT Disim FROM DEPARTMAN WHERE Kat=2;)***	*2.sorgu)* ***Correlative Query*** *SELECT DISTINCT S.Parçaisim* *FROM SATIŞ S* *WHERE S.DepİsimIN* ***(SELECT Disim FROM DEPARTMAN WHERE DepİsimDisim AND Kat=2;)***
3. sorgu) ***EXIST kullanarak*** *SELECT DISTINCT S.Parçaisim* *FROM SATIŞ S* *WHERE EXIST* ***(SELECT * FROM DEPARTMAN WHERE DepİsimDisim AND Kat=2;)***	

Aşağıdaki A ve B sorgularının yaptığı iş aynı mıdır? Eğer değilse fark nedir?

A) SELECT DISTINCT S.ParçaİsimFROM SATIŞ S, DEPARTMAN DWHERE S.DepİsimD.Disim AND Kat <> 2;	B) SELECT DISTINCT ParçaİsimFROM SATIŞWHERE ParçaİsimNOT IN (SELECT DISTINCT ParçaİsimFROM SATIŞ S, DEPARTMAN DWHERE S.DepİsimD.Disim AND Kat = 2);

B) sorgusu 2 katta satılmayan parçaların isimlerini veriyor...

A) sorgusu ise 2. kat dışındaki katlarda satılan parçaları veriyor.(sonuçytaki parça 2. katta da satılıyor olabilir)

İkinci katta bulunan dapartmanlardan en az ikisi tarafından satılan parçaların isimlerini bulan SQL cümlesini yazınız.

```
SELECT DISTINCT S.Parçaisim
FROM SATIŞ S, DEPARTMAN D
WHERE S.DepİsimD.Disim AND Kat=2
GROUP BY S.ParçaİsimHAVING COUNT( DISTINCT D.Disim)>1;
```

```
SELECT S.ParçaİSim
FROM SATIŞ S, DEPARTMAN D
WHERE S.DepİSim=D.Disim AND Kat=2
GROUP BY S.ParçaİSim
HAVING COUNT (DISTINCT D.Disim) = (SELECT COUNT(DISTINCT Disim)
                                   FROM DEPARTMAN
                                   WHERE Kat=2);
```

Yukarıdaki SQL cümlesinin hangi sorguya cevap verdiğini türkçe olarak yazınız.

2. kattaki bütün departmanlar tarafından satılan parçaların isimlerini buluyor..

Bütün işçileri, amirlerinden az kazanan işçileri içeren depatmanların isimlerini bulan SQL cümlesin yazınız..(ipucu: NOT IN kullanabilirsiniz...)

```
SELECT  DISTINCT DepİSim
FROM İŞÇİ
WHERE DepİSim NOT IN
    (SELECT i1. DepİSim
    FROM İŞÇİ i1, İŞÇİ i2
    WHERE i1.amirNo=i2.İno AND i1.maaş >= i2.maaş)
```

Bütün işçileri yöneticisinden az kazanan departmanların isimlerini bulan SQL cümlesini tamamlayın..

```
SELECT D1.Disim
FROM DEPARTMAN D1, İŞÇİ İ1
WHERE D1.YöneticiNo=İ1.İno AND İ1.Maaş > (SELECT Maaş
                                          FROM İŞÇİ İ2
                                          WHERE İ2.DepİSim = D1.Disim AND İ2.İno <> İ1.İno)
```

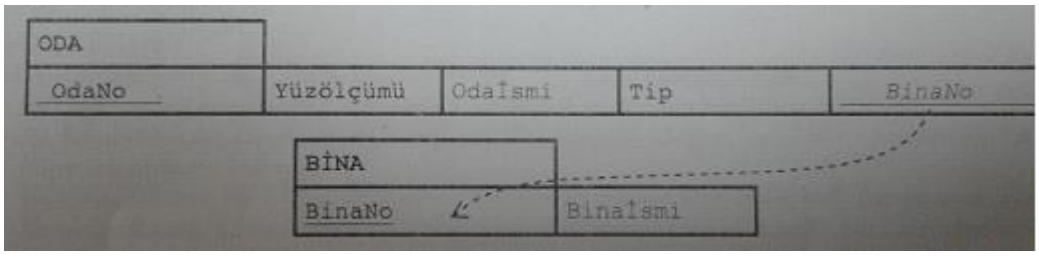
```
SELECT D1.Disim
FROM DEPARTMAN D1, İŞÇİ İ1
WHERE D1.YöneticiNo=İ1.İno AND İ1.Maaş >(SELECT Maaş
                                         FROM İŞÇİ İ2
                                         WHERE İ2.DepİSim = D1.Disim AND İ2.İno NOT IN
                                                (SELECT YöneticiNo FROM
                                                DEPARTMAN)
```

```
CREATE VIEW V1(d1,d2 ) AS
SELECT DepİSim, AVG(Maaş) FROM İŞÇİ
GROUP BY DepİSim;
SELECT İsim, (Maaş – d2) FROM V1, İŞÇİ
WHERE V1.d1 = İŞÇİ.DepİSim
AND İŞÇİ.DepİSim = 'Muhasebe';
```

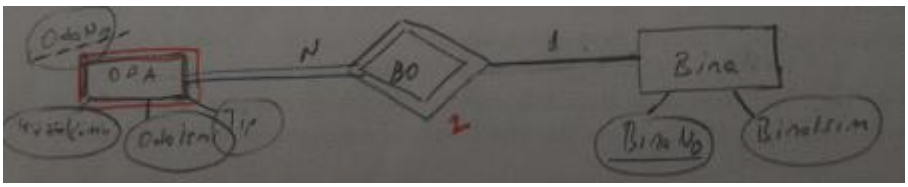
Cümlesi hangi sorguya cevap verir?

Muhasebe departmanındaki işçilerin, her bir işçinin, maaşlarının departmanın ortalama maaşından farkını listeler

27.)



a.) İlişkisel model tabloları yukarıdaki gibi olan veri tabanına ait ER diyagramını çiziniz.



b.) "Elektrik-Elektronik Fak." İsimli binadaki seminer tipindeki odaların numaralarını veren SQL sorgusunu yazınız.

Select o.OdaNo
From Oda o, Bına b
Where o.BinaNo = b.BinaNo AND o.Tip="Seminer" AND b.BinaİSim="EEF"

c.) "Elektrik-Elektronik Fak." İsimli binadaki seminer tipindeki odaların toplam yüzölçümünü veren SQL sorgusunu yazınız.

Select sum(yüzölçüm)
From Oda o, Bına b
Where o.BinaNo = b.BinaNo AND o.Tip="Seminer" AND b.BinaİSim="EEF"

d.) Her binanın yüzölçümünü, BinaNo, Yüzölçümü olarak listeleyen SQL sorgusunu yazınız.

Select o.BinaNo as BinaNo, sum (o.yüzölçüm) as Yüzölçüm
From Oda o
Group by o.BinaNo

e.) "Elektrik-Elektronik Fak." İsimli binadaki en büyük odanın ismini bulan SQL sorgusunu yazınız.

Select o.OdaIsmi
From Oda o
Where o.yüzölçüm = (select max(o.yüzölçüm) from Oda o, Bina b
Where o.BinaIsmi=b.BinaIsmi AND b.BinaIsmi='EEF'

28.) KİTAP tablosundaki bilgiler yandaki gibi olduğuna göre, aşağıdaki soruları cevaplayınız.

ISBN	adı	tür	sayfa sayısı
00001	x	tarih	200
00011	z	bilgisayar	300
00111	y	bilgisayar	100
01111	x	tarih	500
11111	k	fizik	400
10000	l	tarih	200
11000	m	fizik	1000
11100	d	bilgisayar	100
11110	s	kimya	600
00000	s	kimya	200
01010	s	kimya	200
10101	a	tarih	600

```
SELECT k.tür, AVG(k.sayfasayısı)
FROM Kitap k
WHERE k.sayfasayısı>=200
GROUP BY k.tür
HAVING count(*)>2;
```

sorgusunun cevabı: (değer olarak)

tarih 375
kimya 333,333..

```
SELECT Temp.tür, Temp.max
FROM (select k.tür as tür, max(k.sayfasayısı) as max
      from Kitap k
      group by k.tür, k.adı) AS Temp
WHERE Temp.max = (SELECT MAX (Temp.max)
                  FROM Temp);
```

sorgusunun cevabı: (değer olarak)

fizik 1000

DP tablosu:
DNo PNo Miktar
D1 P1 300
D1 P2 200
D1 P3 400
D1 P4 200
D1 P5 100
D1 P6 100
D2 P1 300
D2 P2 400
D3 P2 200
D4 P2 200
D4 P4 300
D4 P5 400

DEPO tablosu:
No Isim Durum Sehir
D1 Ali 20 Istanbul
D2 Cem 10 Izmit
D3 Cenk 30 Izmit
D4 Can 20 Istanbul
D5 Mert 30 Bursa

PARÇA tablosu:
No Isim Durum Agirlik Sehir
P1 Dolap Kirmizi 12 Istanbul
P2 Kapi Yesil 17 Izmit
P3 Masa Mavi 17 Mersin
P4 Masa Kirmizi 14 Istanbul
P5 Perde Mavi 12 Izmit
P6 Sandalye Kirmizi 19 Istanbul

29.) Yukaridaki iliskisel modelde DEPO, PARÇA tablolari yani sira, hangi depodan hangi parçalarin saglandigi bilgisi DP isimli tabloda tutulmaktadir. DP tablosunda Dno DEPO.No'ya, Pno ise PARÇA.No'ya isaret eden "foreign" anahtarlardir.

a.) Masalarin renk ve agirliklarini veren iliskisel algebra sorgusunu yazin.

$$\pi_{durum,agirlik}(\sigma_{isim='masa'}(PARCA))$$

b.) P2 numarali parçayi saglayan depolarin isimlerini veren SQL sorgusunu yazin.

```
1.yol:
select Depo.Isim
from DP, DEPO
where DP.PNo='P2' and DP.DNo=DEPO.No

2.yol:
select Depo.Isim
from DEPO
where No in (
select DNo
from DP
where PNo='P2' )
```

c.) Asagidaki sorgu sonucunu yazin.

```
SELECT Pno, Sum(Miktar)
FROM DP
GROUP BY Pno;
```

P1 600
P2 1000
P3 400
P4 500
P5 500
P6 100

d.) Normalizasyonu inceleyin(10 puan)

Butun tablolar 2NF ve 3NF'yi sagliyor.

30.) Otel (No, Isim, Sehir)

Oda(OdaNo, OtelNo, Tip, Ucret) *foreign anahtar: OtelNo à Otel*

Yukaridaki tablo bir otelin veri tabanina aittir. Buna gore;

a.) Ortalama oda ucreti 200 milyondan dusuk olan otellerin numaralarini veren sql sorgusunu yazin.

```
select OtelNo
from Otel, Oda
where Oda.OtelNo = Otel.No (bu satiri cikartabiliriz,join yapmaya gerek yok)
group by OtelNo
having avg(Oda.Ucret) < 200milyon
```

```
create view myview(OtelNo,ortalama)
as (select OtelNo, avg(Ucret) from Oda
group by OtelNo)
select OtelNo from myview where ortalama < 200milyon
```

31.) Öğrenci (isim, no, gpa, danışman) *FK: danışman → Öğretmen.isim*
Öğretmen (isim, bölüm)

Yukarıdaki tablolar öğrenci ve öğretmenler hakkında bilgi tutan bir veritabanına aittir.
'BBM' bölümündeki bütün öğrencilerin isimlerini bulan aşağıdaki sorgu yazılıyor. Bu sorgu doğru mudur? **Eğer yanlış ise** doğrusunu yazınız.

```
YANLIS. Cunku ic sorgu sutun gonderir. Esitligin sag
tarafinda kullanilmaz.
1. YOL:  select isim
         from Öğrenci
         where danışman IN (select isim
                            from Öğretmen
                            where bölüm = 'BBM')
2. YOL:  select isim
         from Öğrenci, Ogretmen
         where danışman=Ogretmen.isim AND
               bölüm = 'BBM')
```

```
select isim
from Öğrenci
where danışman = (select isim
                  from Öğretmen
                  where bölüm = 'BBM')
```

b.) Personel (no, isim, yaş, maaş)
Yukarıdaki tablo personel veri tabanına ait bir tablodur.
Ortalama personel maaşından düşük olan personelin maaşına %10 artış yapan aşağıdaki sorgu yazılıyor.
Bu sorgu doğru mudur? **Eğer yanlış ise** doğrusunu yazınız.

```
update Personel
set maaş = 1.1 * maaş
where maaş < ( select avg(maaş) from Personel )
```

```
YANLIS. Cunku update hem ilgili satiri hem de `where` deki kosulu etkiliyor.
//no PK olsun.
create view X(no,maas) as select no,maas
                          from Personel
                          where maas< (select avg(maas)
                                       from Personel)
update X
set maas = 1.1*maas
-----------------------------------------------
NOT: Asagidaki cozum YANLIS, yukarida belirtilen problem devam ediyor.
create view X(ort_maas) as select avg(maas)
                           from Personel
update X
set maas = 1.1*maas where maas< (select ort_maas from X)
```

32.)

```
create table Kitap    (ISBN integer Primary Key);
create table Kitapçı (ID integer Primary Key, İsim Char(20), Adres Char(30) );
create table Sahip    (ID integer, ISBN char(10),
                       Primary key (ID, ISBN),
                          Foreign key (ID) references Kitapçı (ID),
                          Foreign key (ISBN) references Kitap (ISBN));
```

```
select İsim
from Kitapçı
where id = ( select S1.id
             from Sahip S1
             group by S1.id
             having count(*) > all
              (
                select count(*)
                from Sahip S2
                where S1.id != S2.id
                group by S2.id
              )
            );
```

Verilen tablolar için, yukarıdaki SQL sorgusunun ne yaptığını 1 cümlede Türkçe olarak yazınız.

En fazla kitaba sahip olan kitapçının ismini bulur